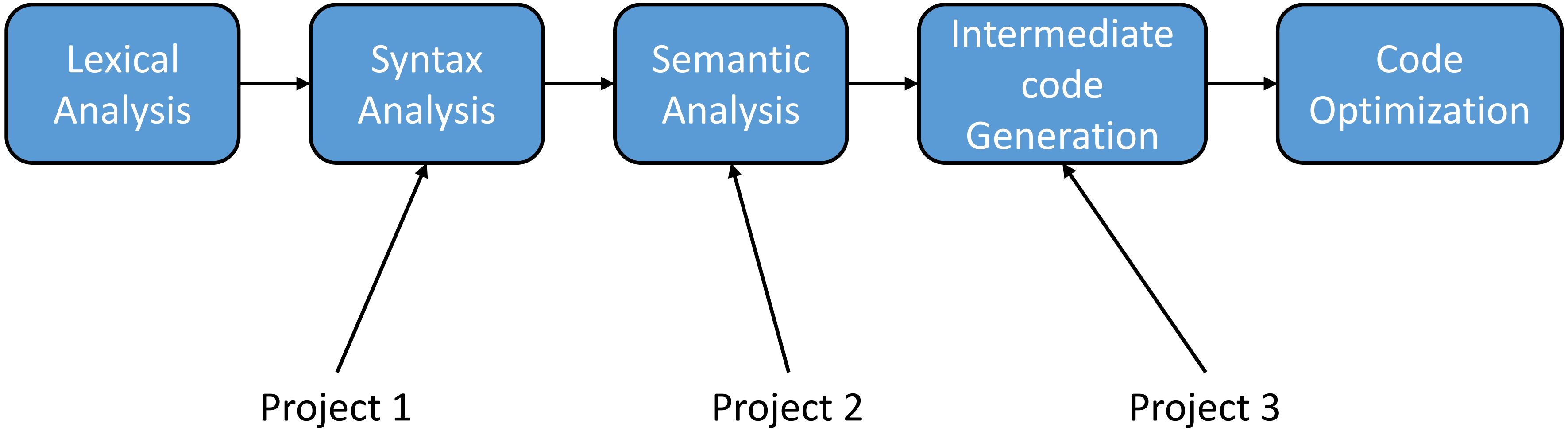
Intro. To Compiler Construction

Instructor: Feng-Jian Wang

(王豐堅)

Projects Overview



Program Assignment 1

- Implement a parser for the MiniJava language in appended part I, you may use JavaCC or SableCC. **Due: Nov. 22th**

Input file

```
Class Factorial {
    Public static void main ( String[] a) {
            System.out.println(new Fac().ComputerFac(10));
         }
    }
    Class Fac {
        Public int ComputerFac(int num) {
            Int num_aux;
            If (num < 1)
                Num_aux = 1;
            Else
                Num_aux = num * (this.ComputeFac(num-1));
                Return num_aux;
    }
}
```

Output file

```
exp -> new Fac()
exp -> 10
expRests ->
expList ->
exp -> exp ComputeFac(expList)
statement -> System.out.println( exp ) ;
.
.
.
.

Error(s):
.
.
```

Program Assignment 1

- How to submit?
  - Upload your *.java code.
  - Input and output files.
  - Upload a report including:
    - How to run the program.
    - Extra point if you could explain the code.
  - Compress the files and name the compressed file using this following format: student ID_name_project1, e.g. 0780807_費和民_project1
- Grading Policy:
  - TA will test the program using a Factorial.java file only.
  - TA may add a simple error into the file to check whether your program could find it.
  - If you submit your program and report, even it cannot be compiled, you will get at least 60.
  - If the program can find an error based on the grammar, e.g., string error, you will get 90.
  - + simple explanation of the code -> 10 points.

Program Assignment 2

- Design a set of visitors which type-checks a MiniJava program and produces any appropriate error messages about mismatching types or undeclared identifiers. **Due: Dec. 13th**

Input file

```
Class Factorial {
  Public static void main ( String[] a) {
      System.out.println(new Fac().ComputerFac(10));
    }
  }
  Class Fac {
    Public int ComputerFac(int num) {
      Int num_aux;
      If (num < 1)
        Num_aux = 1;
      Else
        Num_aux = num * (this.ComputeFac(num-1));
        Return num_aux;
  }
}
```

Output file

```
Classes:
  Fac
    Fields:
    Methods:
      int ComputeFac
        Params:
          int num
        Locals:
          int num_aux
.
.
.
Type-checking successful.
```

Program Assignment 2

- How to submit?
  - Upload your *.java code.
  - Input and output files.
  - Upload a report including:
    - How to run the program.
    - Extra point if you could explain the code.
  - Compress the files and name the compressed file using this following format: student ID_name_project1, e.g. 0780807_費和民_project2
- Grading Policy:
  - TA will test the program using a Factorial.java file only.
  - TA may add a simple error into the file to check whether your program could find it.
  - If you submit your program and report, even it cannot be compiled, you will get at least 60.
  - If the program can find an error based on the grammar, e.g., Type mismatch, you will get 90.
  - + simple explanation of the code -> 10 points.

Program Assignment 3

- Implement a simpler Translator with a set of visitors, to translate a MiniJava program into intermediate representation trees. **Due: Jan. 5th**

Input file

```
Class Factorial {
    Public static void main ( String[] a) {
            System.out.println(new Fac().ComputerFac(10));
         }
    }
    Class Fac {
        Public int ComputerFac(int num) {
            Int num_aux;
            If (num < 1)
                Num_aux = 1;
            Else
                Num_aux = num * (this.ComputeFac(num-1));
                Return num_aux;
    }
}
```

Output file

```
Intermediate code for main:
EXPR(
  CALL(
    NAME printInt,
       MEM(
         BINOP(PLUS,
             CALL(
                 NAME Fac$ComputerFac,
                    CALL(
.
.
.
Intermediate code for Fac$ComputerFac:
.
.
.
```

Program Assignment 3

- How to submit?
    - Upload your *.java code.
    - Input and output files.
    - Upload a report including:
        - How to run the program.
        - Extra point if you could explain the intermediate representation trees.
        - Extra point if you could explain the code.
    - Compress the files and name the compressed file using this following format:
      student ID_name_project1, e.g. 0780807_費和民_project3
- Grading Policy:
    - TA will test the program using a Factorial.java file only.
    - If you submit your program and report, even it cannot be compiled, you will get at least 60.
    - If the program could produce intermediate representation tree for the sample program (Factorial.java), you will get 90.
    - + explanation of the produced intermediate representation trees.
    - + simple explanation of the code -> 10 points.